ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method · Nashville, Tennessee · April 19–23, 2015, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2015)

SimpleMOC - A PERFORMANCE ABSTRACTION FOR 3D MOC

Geoffrey Gunow, John Tramm, Benoit Forget, and Kord Smith
Department of Nuclear Science & Engineering
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139
geogunow@mit.edu; jtramm@mit.edu; bforget@mit.edu; kord@mit.edu

Tim He
Center for Exascale Simulation of Advanced Reactors
Argonne National Laboratory
9700 South Cass Avenue, Lemont, IL 60439
shuohe@anl.gov

ABSTRACT

The method of characteristics (MOC) is a popular method for efficiently solving two-dimensional reactor problems. Extensions to three dimensions have been attempted with mitigated success bringing into question the ability of performing efficient full core three-dimensional (3D) analysis. Although the 3D problem presents many computational difficulties, some simplifications can be made that allow for more efficient computation. In this investigation, we present SimpleMOC, a "mini-app" which mimics the computational performance of a full 3D MOC solver without involving the full physics perspective, allowing for a more straightforward analysis of the computational challenges. A variety of simplifications are implemented that are intended to increase the computational feasibility, including the formation axially-quadratic neutron sources. With the addition of the quadratic approximation to the neutron source, 3D MOC is cast as a CPU-intensive method with the potential for remarkable scalability on next generation computing architectures.

Key Words: Method of Characteristics, High Performance Computing

1 INTRODUCTION

Over the last few decades computational power has grown exponentially with growing transistor count according to Moore's Law [1]. This has allowed previously intractable computational problems to be investigated. For reactor physics simulations, this means the long time goal of achieving full-core three-dimensional (3D) transport reactor simulations is now within reach. While two-dimensional assembly and core transport calculations have "reached industrial maturity" [2], they suffer from a loss of accuracy when compared with three-dimensional models. For high fidelity modeling, the third dimension is necessary to correctly predict neutron leakage as well as axial power distributions in heterogeneous reactors. One attractive method for three-dimensional modeling is the Method of Characteristics (MOC). This method is commonly used in two-dimensional neutron transport simulations and is naturally extensible to three dimensions.

While MOC is easily extensible to three dimensions conceptually, the implementation of such an algorithm is difficult in practice as the computational requirements can be overwhelming [3]. This motivates the development of a "mini-app" named SimpleMOC which replicates the computational performance of a full 3D MOC solver but lacks elements necessary to achieve full solutions. For instance, SimpleMOC mimics access patterns likely to be cache misses by randomly selecting locations in memory from which to load data. This simplified application allows for much easier analysis of computational performance.

2 SimpleMOC PHYSICS ASSUMPTIONS

For accurate replication of the 3D MOC computational performance, assumptions need to be made about how the full 3D MOC solver would be structured. In this investigation, we make a variety of assumptions intended to increase computational performance. These assumptions include axially extruded geometries, on-the-fly axial ray tracing, and axially-quadratic sources.

2.1 Geometrical Simplifications

To reduce the memory and CPU requirements associated with 3D ray tracing, the geometry is assumed to be axially extruded. This means that one radial plane is formed which is a superposition of all radial detail. An example is shown in Fig. 1.

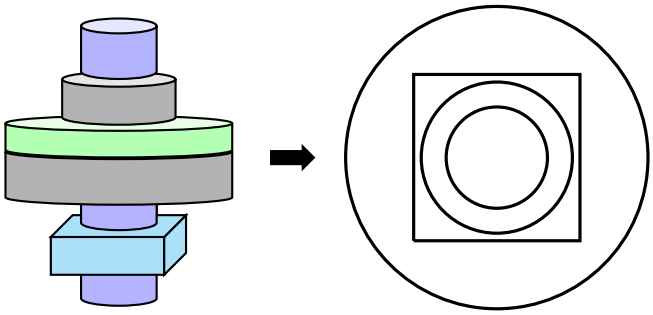


Figure 1. Depiction of extruded geometry and compression of the 3D geometry into a radial plane which is formed by a superposition of all radial geometrical information.

Once this radial plane is formed, all axial ray tracing can be decoupled from the radial ray tracing. This allows for common ray tracing and storage of track segment lengths to be conducted on the radial plane using the same ray tracing schemes employed for standard 2D MOC implementations. For SimpleMOC, a pre-computed 2D ray tracing file is loaded from an external application, such as OpenMOC [4]. During the computation of angular flux attenuation, on-the-fly ray tracing is conducted in the axial direction using only a 1D grid, which involves trivial computational resources.

Each loaded 2D track is used to create multiple 3D tracks. First, the 2D track is projected in the axial direction at multiple polar angles $\theta \in (0, \pi)$. Then for each polar angle, a set of tracks is created with some axial separation to fill the entire axial height, forming set of "z-stacked" tracks. This ensures that the tracks span the entire polar and axial spaces. During transport sweeps, each set of z-stacked rays is traversed until all tracks in the set have either completed all 2D segments or have exited through the top or bottom surfaces. When a track encounters a boundary defined by the simple 1D axial grid, the 2D segment length is shortened by 2D distance traversed in the current region and then continued until completion or intersection with a domain boundary.

2.2 Axially-Quadratic Sources

Due to the structure of common reactor designs allowing for the success of the popular 2D/1D transport methods [5–7], it is believed that over large axial regions (on the order of 15 cm) the cross-sections can be approximated as being constant. For the 2D/1D methods to have any reasonable success, this must be true, as 2D/1D methods only weakly incorporate axial detail. Constant cross-section axial regions including neutron source information would have the capability to include far more axial information.

In each constant cross-section region, the neutron source is approximated as quadratic in the axial direction. This is motivated by past success of implementing quadratic source approximations in nodal methods [8]. Most MOC implementations rely on the use of flat source regions over which the neutron source and cross-sections are assumed to be constant [9]. Common implementations of higher order sources involve tallying flux moments during the transport sweep iterations [10]. After each transport sweep, new sources are calculated for the next transport sweep by first reconstructing the scalar flux and then calculating the contribution from each group flux to all group sources. This calculation has a computational complexity of $\Theta(NG^2)$ where N is the number of source regions and G is the number of energy groups. For a high number of energy groups, this computation can become cumbersome. This is troubling since high fidelity calculations require on the order of 100 energy groups.

Therefore, SimpleMOC employs an alternative strategy. Taking advantage of the fact that large coarse axial regions can be approximated as having constant cross-sections, each coarse axial region is split into several fine axial regions. The source contribution in each fine axial region can be directly formed on-the-fly during the transport sweeps with the source strength calculated by fitting a quadratic function to the neighboring fine axial regions. To simplify the process and lessen computational requirements, the fitting routine only involves the fine axial region being traversed and the neighboring fine axial regions. Since three points uniquely defines a quadratic function, the function can be directly computed without any fitting routine. This process is illustrated in Fig. 2. It is important to note that only fine axial regions from the current coarse axial region are used to construct the source, guaranteeing that material properties are constant across the extent of the fit.

If a given reactor is 400 cm in axial length with cross-sections reasonably constant over axial regions of 15 cm, roughly 30 coarse axial regions would be required for each 2D region. The coarse axial intervals would be divided into sufficiently many fine axial regions to converge the source.

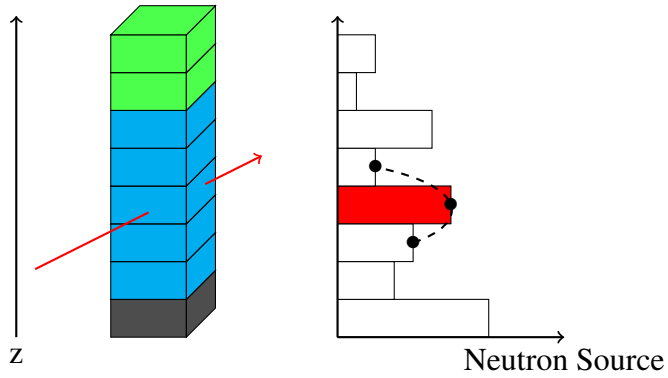


Figure 2. Depiction of quadratic source computation in which the stack on the left represents fine axial regions colored by constant cross-section values. The plot on the right depicts on-the-fly quadratic source computation.

With the assumption of 5 fine axial regions per coarse axial, this results in 150 fine axial regions per 2D region across which fluxes and sources are tallied.

2.3 Quadratic Source Computation

With the implementation of higher order axial sources, the number of floating point operations (FLOPs) per angular flux intersection greatly increases. To analyze the computational performance, we first need to consider the MOC equations over the constant cross-section fine axial regions across which the equations are applied. We are primarily concerned with two computations: the attenuation of the angular flux ψ for a particular energy group along a track and the contribution of that angular flux to the scalar flux ϕ of the fine axial region. To determine the angular flux attenuation, ψ is defined as a function of distance s traversed along the track in the fine axial region with total cross-section Σ_t. Whereas the normal approximation of a constant source q_0 leads to simple exponential attenuation given by Eq. 1, the angular flux attenuation with a quadratic source approximation leads to a much more computationally demanding equation.

$$\psi(s) = \psi(0)e^{-\Sigma_t s} + \frac{q_0}{\Sigma_t}\left(1 - e^{-\Sigma_t s}\right) \tag{1}$$

To derive the equations for a quadratic source approximation, a definition for the quadratic source first needs to be defined. The neutron source q as a function of axial distance z is given by Eq. 2.

$$q(z) = c_0 + c_1 z + c_2 z^2. \tag{2}$$

To solve the transport equation, the source needs to be presented as a function of traversed distance s. This requires a change of variables presented in Eq. 3 with θ representing the polar angle and z_{in} representing the axial height of the track entering the fine axial region. This yields the source as function of s presented in Eq. 4.

$$z = s\cos\theta + z_{\text{in}} \tag{3}$$

$$\begin{aligned} q(s) &= \left(c_0 + c_1 z_{\text{in}} + c_2 z_{\text{in}}^2\right) + \left(c_1 + 2c_2 z_{\text{in}}\right)\left(s\cos\theta\right) + c_2\left(s\cos\theta\right)^2 \\ &= q_0 + q_1 s + q_2 s^2 \end{aligned} \tag{4}$$

Remembering that the integral form of the transport equation in the MOC formulation can be presented as

$$\psi(s) = \psi(0)e^{-\Sigma_t s} + \int_0^s ds' q(s') e^{-\Sigma_t (s-s')}, \tag{5}$$

the resulting angular flux attenuation equation for quadratic sources is given by

$$\begin{aligned} \psi(s) = \psi(0)e^{-\Sigma_t s} &+ \frac{q_0}{\Sigma_t}\left(1 - e^{-\Sigma_t s}\right) + \frac{q_1\cos\theta}{\Sigma_t^2}\left(\Sigma_t s + e^{-\Sigma_t s} - 1\right) \\ &+ \frac{q_2\cos^2\theta}{\Sigma_t^3}\left(\Sigma_t s\left(\Sigma_t s - 2\right) + 2\left(1 - e^{-\Sigma_t s}\right)\right). \end{aligned} \tag{6}$$

The scalar flux ϕ can be calculated using the angular flux attenuation of all tracks crossing the region. Eq. 7 shows the calculation of the scalar flux ϕ as a weighted sum of the average angular flux $\bar{\psi}_k$ for all tracks k that traverse a distance l_k across the region. The track weights are denoted w_k and the volume of the region is V.

$$\phi = \frac{4\pi}{V}\sum_{k\in V} w_k l_k \bar{\psi}_k. \tag{7}$$

For the flat source approximation, the average angular flux $\bar{\psi}$ for a given track can be calculated by Eq. 8 where $\psi(0)$ is the angular flux of a track entering the region and $\psi(l)$ is the angular flux leaving the region after traversing a distance l.

$$\bar{\psi} = \frac{\psi(0) - \psi(l)}{l\Sigma_t} + \frac{q_0}{\Sigma_t} \tag{8}$$

This is simple to calculate as the angular flux attenuation $\psi(0)-\psi(l)$ has already been calculated in Eq. 1 and the constant q/Σ_t term can be added to each scalar flux tally after the transport sweep. For quadratic sources, the calculation of the average angular flux is not as simple, with the required calculation given in Eq. 9.

$$\begin{aligned}\bar{\psi} = \frac{1}{l}\Bigg[&\frac{1}{\Sigma_t^2}\left(q_0\Sigma_t l + \left(\Sigma_t\psi(0) - q_0\right)\left(1-e^{-\Sigma_t l}\right)\right) \\ &+\frac{q_1\cos\theta}{2\Sigma_t^3}\left(\Sigma_t l\left(\Sigma_t l - 2\right) + 2\left(1-e^{-\Sigma_t l}\right)\right) \\ &+\frac{q_2\cos^2\theta}{3\Sigma_t^4}\left(\Sigma_t l\left(\Sigma_t l\left(\Sigma_t l - 3\right)+6\right) - 6\left(1-e^{-\Sigma_t l}\right)\right)\Bigg]\end{aligned} \tag{9}$$

Notice that in both the computation of the angular flux attenuation and average scalar flux for quadratic sources, there are three exponential terms instead of just one, as is the case with the flat source approximation. With efficient computation, this fact alone does not add much computational work as the exponential only needs to be computed once and then reused. However, there is a significant increase in the number of multiplications, causing a large increase in the required number of FLOPs. Due to the length of the equation, there is also increased register pressure as the number of values needed to be stored exceeds the space available in the registers. Therefore, great care should be taken in efficiently computing the attenuation and contribution to the scalar flux.

3 IMPLEMENTATION & PERFORMANCE RESULTS

3.1 SimpleMOC Application

The SimpleMOC application is an Open Source software project available online [11]. It is written in C, using MPI for inter-node distributed memory communication and OpenMP for on-node shared memory threading.

Storing flux data for all 3D tracks across the full reactor is impractical for a single computer to handle due to memory requirements and reasonable time-to-solution constraints [12]. Therefore, the problem is domain decomposed to many nodes in a distributed memory fashion using MPI. This allows for each node in the problem to only handle a small portion of the geometry of the reactor, greatly reducing the computational and memory burdens of each node. However, domain decomposition also adds in the burden of storing and exchanging boundary flux data for each characteristic track in the calculation. This new requirement significantly increases the memory footprint requirement of each node, but also allows for much larger problems to be handled than would be tractable by a single node.

This section investigates the performance characteristics of the SimpleMOC application when run on a variety of computing architectures. The performance scaling of both single node archi-

tectures (thread performance) and many node systems (communication costs) are presented and discussed.

3.2 Multi-Node Results

For our target problem, the BEAVRS benchmark [13], we estimate that our formulation of the 3D MOC transport algorithm, as represented by SimpleMOC, requires approximately 75 TB of data when distributed across multiple nodes, assuming the parameters given in Table I. This is primarily (over 94%) composed of boundary flux data that is computed and then exchanged between neighboring nodes after each sweep until convergence of the problem solution. As the amount of data exchanged between sweeps is significant (nearly the full memory available to each node between each sweep), there exists the potential for significant communication penalty if the computational time required for attenuating neutron fluxes through the characteristic lines of the domain is very small. To investigate these concerns, a weak scaling study (where the problem size per node remains constant throughout scaling) was performed with the SimpleMOC application on the IBM Blue Gene/Q Supercomputer *Mira* at Argonne National Laboratory, as shown in Fig. 3. This weak scaling study represents a problem size per node of approximately 970 million geometrical track segments to process and 12.5 GB of boundary flux data to exchange for a single sweep – corresponding to the workload per node of the full BEAVRS benchmark when distributed across 5,780 nodes. These results show that the algorithm scales extremely well through 512 nodes, achieving a stable wall time despite the increase in overall total problem size as nodes were added.

Table I. Anticipated parameters for a converged 3D MOC solution of a PWR core.

Parameter	Dimension
Height (z)	400 cm
Assembly width (x and y)	21.24 cm
Geometry layout	17×17 assemblies
Radial ray spacing	0.05 cm
Axial ray spacing	0.25 cm
Number of azimuthal angles	64
Number of polar angles	10
Number of energy groups	100
Axial source spacing	10 cm
Fine axial regions per source	5

Further investigation was performed by implementing wall-time counters into the SimpleMOC source code to indicate the amount of time spent in each particular phase of the computation (e.g., flux attenuation, global statistical reductions, border flux communication exchange). This allowed us to determine the communication overhead of the computation, and whether or not there was sufficient work to be performed between communication rounds to ensure that our algorithm is not communication bound. As shown in Fig. 3, we can see that the runtime of the calculation is in fact dominated by the core computational task of the algorithm (i.e., attenuation of neutron fluxes along a track). Only 2.3% of the runtime is spent on data exchange between nodes, a ratio which remains constant for all configurations that we examined. These results indicate that our implementation of

the 3D MOC algorithm is able to efficiently map onto many node systems without worry of any significant communication overhead.

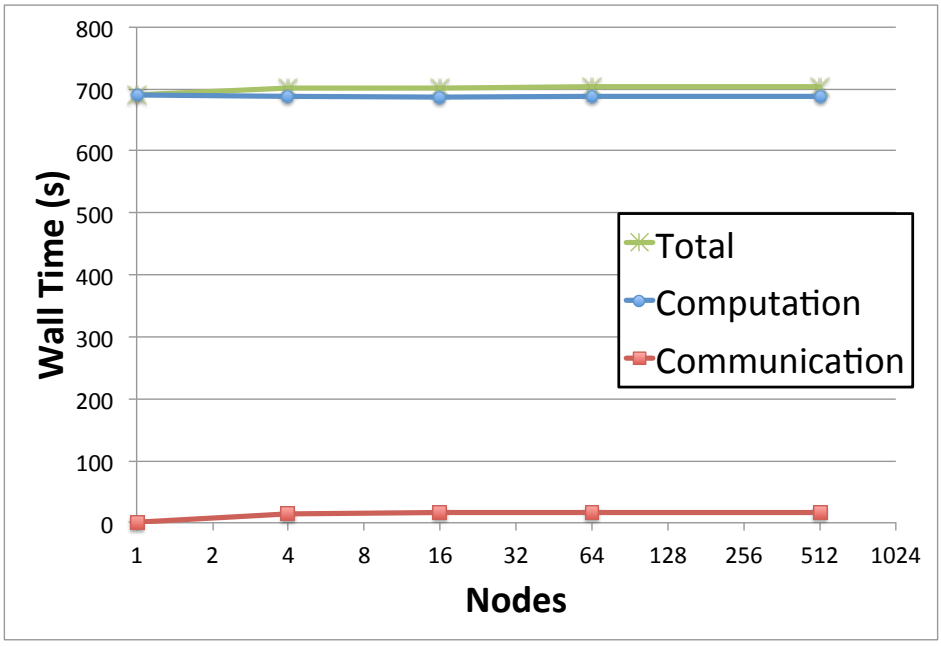


Figure 3. SimpleMOC Weak Scaling on IBM Blue Gene/Q *Mira*, up to 512 nodes

3.3 Single Node Results

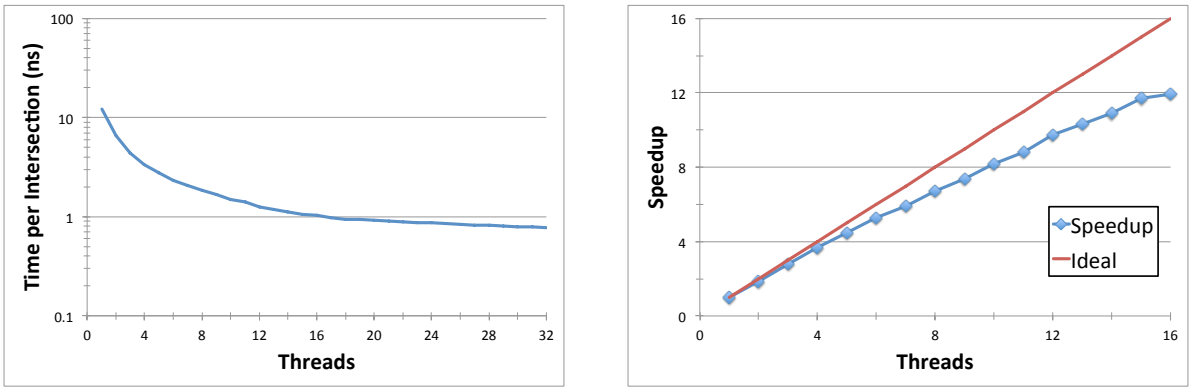
Understanding the single node, shared memory performance characteristics of an algorithm is key to predicting time-to-solution of a full application as well as performance of an application on future high performance computing (HPC) architectures. For instance, applications that are primarily memory bandwidth bound or experience frequent computational stalls due to thread contention will likely be poor candidates for next generation architectures, as the number of cores per node is multiplying at a much faster rate than bandwidth and latency masking resources. Applications that feature high compute intensities (i.e., the number of floating point operations per memory load) and are able to be expressed in terms of wide SIMD vector operations will run very efficiently on next generation HPC architectures.

3.3.1 Strong Scaling

To determine the performance characteristics of our formulation of the 3D MOC algorithm, we performed strong scaling studies on two current generation HPC systems: a single node of the IBM Blue Gene/Q supercomputer *Mira* featuring16 cores (64 threads), and a single node of a dual socket Xeon E5-2650 system with 16 cores (32 threads). Furthermore, we instrumented the SimpleMOC application using PAPI performance counters in order to gain deeper insight into hardware resource usage and to determine where the bottleneck occurs in the application [14]. The figure of merit in SimpleMOC is the *time per intersection*, which indicates the amount of time spent computing the

attenuation factors and quadratic source contributions of a single energy neutron flux across a single geometric segment of a characteristic track within the reactor.

3.3.2 Xeon Node Results



(a) Strong scaling **(b) Speedup**

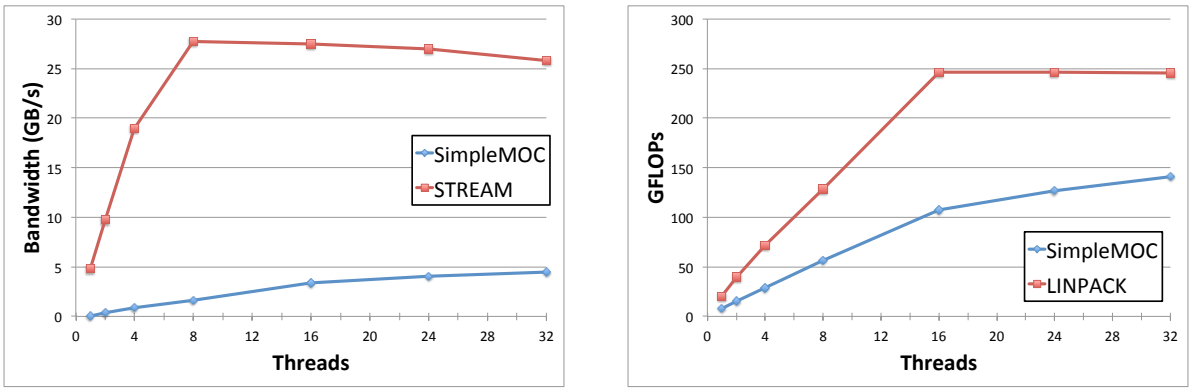
Figure 4. SimpleMOC performance scaling on 16-core Xeon Node

Fig. 4a and Fig. 4b show that the code scales well out to the maximum number of cores on the system. When hardware threads are also added allowing for 2 threads per core for the Xeon system, we find that performance increases significantly. Gains from hardware threading were significant, showing a 34% speedup over only running a single thread per core. Our code was able to achieve a time per intersection of 12.3 ns for a single thread, and 0.77 ns when running all 32 threads on the Xeon node.

3.3.3 Hardware Resource Usage

We were able to deduce a number of parameters from the PAPI performance counters as well, such as floating point operations per second and main memory bandwidth usage. Fig. 5a shows clearly that our application uses only a very small percentage of the maximally available bandwidth of the system (as compared to the STREAM benchmark [15], which represents the maximal bandwidth usage attainable by a normal user application). Fig. 5b shows the floating point resource usage of SimpleMOC when compared to LINPACK, which is a standard benchmark for determining the maximum achievable floating point performance of a user application on a given system [16]. We can see that SimpleMOC is in fact using a huge percentage of the peak FLOP resources available on the system, nearly 60% of the FLOP performance that the LINPACK benchmark achieves. This is an incredibly high number, as most HPC applications often only achieve 5-10%.

The excellent FLOP utilization of the SimpleMOC application can be attributed to the fact that the inner portion of the kernel (which composes 97.7% of the runtime of the application) is composed of several computationally intense loops over all energy groups in the simulation. As our benchmark reactor problem tracks 100 energy groups, and these inner loops can be expressed in an



(a) Bandwidth (GB/s) usage **(b) Floating point performance**

Figure 5. SimpleMOC performance profile on 16-core Xeon Node

easily vectorizable nature, the compiler is able to vectorize these loops using SIMD instructions to take advantage of the vector widths of the floating point units of both the Blue Gene/Q and Xeon architectures. To achieve performance of this level, the source code was aggressively optimized to ensure that all inner loops were properly vectorized by the compiler.

This highlights the importance to the performance of the algorithm of placing energy groups as the inner loop of the computation. Several other configurations are possible, such as placing the polar angle in the inner loop, but experimentation revealed that energy groups served the role far better due to their higher loop iteration count (100 energy groups vs. only 10 polar angles, for our target problem). This is due to the slight overhead of using vector instructions as compared to scalar instructions, with the vector instructions not paying dividends over scalar until there are usually more than 16 to 32 iterations in the loop.

3.3.4 Performance Sensitivity to Problem Size

In order to determine the impact of problem size on performance we varied the axial ray spacing and measured both the sweep time and the time per intersection. As shown in Table II, overall sweep time for the calculation was directly related to axial ray spacing, indicating that there is no performance loss due to variations in problem size. The time per intersection remained constant at 0.77 ns, further demonstrating that SimpleMOC is able to achieve good performance regardless of problem size.

Table II. Axial Ray Spacing Impact on Sweep Time and Performance

Axial Ray Spacing (cm)	Sweep Time (s)	Time per Intersection (ns)
1.0	19.73	0.77
0.5	38.79	0.77
0.25	76.96	0.77
0.125	152.79	0.77

3.3.5 Blue Gene/Q Node Results

Fig. 6 shows the strong scaling of SimpleMOC on a single node of the Blue Gene/Q supercomputer. Using all 64 threads on the node SimpleMOC was able to achieve a time per intersection of 7.11 ns. This result shows that single node performance was 9.2 times slower on the Blue Gene/Q architecture when compared to the Xeon system, which is expected because of the large power usage differential between the two systems. A Blue Gene/Q CPU uses 31.25 W under floating point conditions running all 64 threads [17] compared to the rated combined thermal design power of 190 W for two Xeon E5-2650 chips [18]. This power differential is widened further by the Xeon's higher DRAM power costs as well as larger board and power supply losses. Speedup is not shown for the BG/Q architecture as the architecture heavily relies on hardware threading due to its lack of an out of order execution pipeline. This can lead to misleading results.

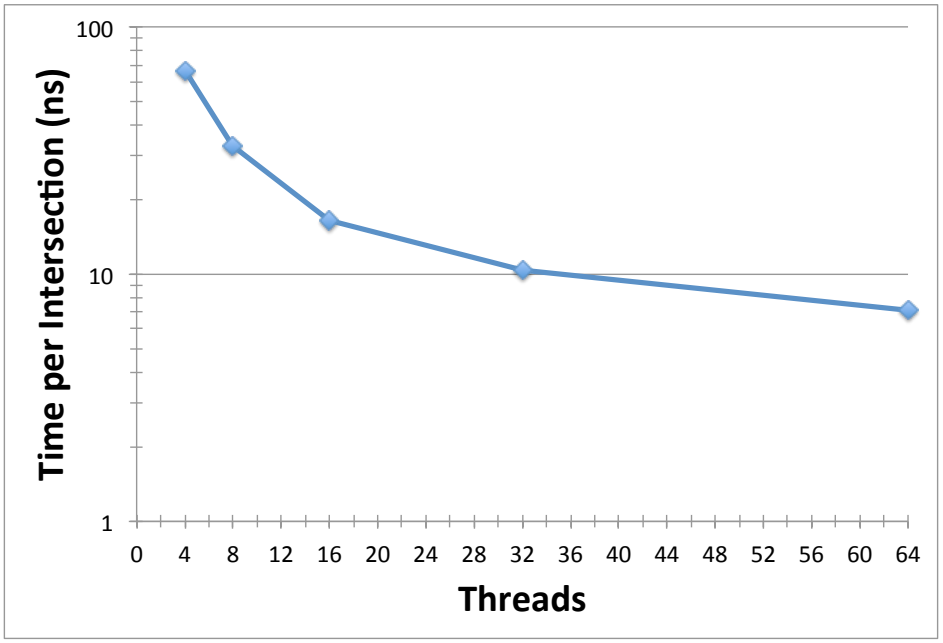


Figure 6. SimpleMOC Strong Scaling on 16-core node of Blue Gene/Q

3.4 Scaling Projections on Future Generation Computer Architectures

Next generation high performance computation architectures are likely to feature huge numbers of processing cores per node and wide SIMD vector floating point units [19–22]. Both of these trends meet the mandate to increase the overall floating point capacity of systems without increasing the amount of power required. Furthermore, these next generation systems are likely to only marginally increase their available bandwidth to main memory, which is expensive both economically and from a power standpoint, highlighting the extreme importance for algorithms to achieve high compute intensities.

In this light, our formulation of the 3D MOC algorithm, as implemented in SimpleMOC, should map extremely well to architectures of the future due to its ability to utilize wide SIMD vector

units as well as having minimal on-node bandwidth requirements. Additionally, the inter-node communication costs of the algorithm are very low, allowing the algorithm to scale to huge numbers of nodes with little overhead.

4 CONCLUSIONS

Three-dimensional solutions of the neutron transport equation have recently become feasible, allowing for higher fidelity modeling of neutron physics. One method available to solve the neutron transport equation is MOC. Here we have demonstrated a mini-app named SimpleMOC which replicates the performance of an optimized 3D MOC solver under a variety of assumptions intended to boost computational performance, including axially-quadratic sources.

With the addition of the quadratic sources, SimpleMOC performance is shown to highly utilize the floating point capacity. With trends in the development of next generation high performance computing architectures placing high importance on utilization of floating point capacity, SimpleMOC shows great potential for future scalability. Several key results are highlighted in Table III, showing that the SimpleMOC application is capable of excellent scaling on many-node supercomputing architectures as well as extremely high resource utilization of each node.

Table III. Summary of Key Results

Blue Gene/Q Inter-node Communication Costs (512 Nodes)	2.3% of Runtime
Blue Gene/Q Time per Intersection	7.11 ns
Xeon Time per Intersection	0.77 ns
Xeon Bandwidth Usage	16% of System
Xeon FLOP Usage	57% of System

5 ACKNOWLEDGMENTS

The first author is a recipient of the U.S. Department of Energy Office of Nuclear Energy's Nuclear Energy University Programs Fellowship. Additional support was provided by the Center for Exascale Simulation of Advanced Reactors (CESAR), a co-design center under the U.S. Department of Energy . The authors would also like to thank Michael Smith from Argonne National Laboratory for providing guidance from his experience with 3D MOC solvers.

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract DE-AC02-06CH11357 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

6 REFERENCES

[1] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114–117, 1965.

[2] R. Sanchez. Prospects in deterministic three-dimensional whole-core transport calculations. *Nuclear Engineering and Technology*, 44(2):113–150, 2012.

[3] B. Kochunas. A Hybrid Parallel Algorithm for the 3-D Method of Characteristics Solution of the Boltzmann Transport Equation on High Performance Computing Clusters. Ph.D. Thesis, University of Michigan, Department of Nuclear Engineering and Radiological Sciences (2013).

[4] William Boyd, Samuel Shaner, Lulu Li, Benoit Forget, and Kord Smith. The OpenMOC Method of Characteristics Neutral Particle Transport Code. *Annals of Nuclear Energy*, 2014.

[5] Y. S. Jung, et al. Practical numerical reactor employing direct whole core neutron transport and subchannel thermal/hydraulic solvers. *Annals of Nuclear Energy*, 62:357.

[6] H.G. Joo, et al. Methods and performance of a three-dimensional whole-core transport code DeCART. In *PHYSOR*, Chicago, IL, USA, 2004.

[7] B. W. Kelley and E. W. Larsen. 2D/1D approximations to the 3D neutron transport equation. I: Theory. In *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, Knoxville, TN, USA, 2013.

[8] J. Cronin, et al. SIMULATE-3 Methodology Manual. Studsvik of America, Inc., 1995.

[9] W. Boyd. Massively Parallel Algorithms for Method of Characteristics Neutral Particle Transport on Shared Memory Computer Architectures. M.S. Thesis, Massachusetts Institute of Technology, Department of Nuclear Science and Engineering (2014).

[10] R. Ferrer, J. Rhodes, and K. S. Smith. Linear Source Approximation in CASMO-5. In *PHYSOR*, Knoxville, TN, USA, 2012.

[11] Geoff Gunow and John Tramm. SimpleMOC: A mini-app to represent the 3D method of characteristics algorithm for neutron transport. `https://github.com/ANL-CESAR/SimpleMOC`, 2014.

[12] S. Shaner, et al. Theoretical Analysis of Track Generation in 3D Method of Characteristics. Submitted to the *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, 2015.

[13] Nicholas Horelik, Bryan Herman, Benoit Forget, and Kord Smith. Benchmark for evaluation and validation of reactor simulations (BEAVRS). *International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)*.

[14] Innovative Computing Laboratory. PAPI - performance application programming interface. `http://icl.cs.utk.edu/papi/index.html`, 2013.

[15] John D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, 1995.

[16] Jack Dongarra and Piotr Luszczek. Linpack benchmark. In David Padua, editor, *Encyclopedia of Parallel Computing*, pages 1033–1036. Springer US, 2011.

[17] K. Yoshii, K. Iskra, R. Gupta, P. Beckman, V. Vishwanath, Chenjie Yu, and S. Coghlan. Evaluating power-monitoring capabilities on IBM Blue Gene/P and Blue Gene/Q. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 36–44, Sept 2012.

[18] Intel. Xeon processor E5-2650 CPU specifications. `http://ark.intel.com/products/64590/`, 2013.

[19] S.S. Dosanjh, R.F. Barrett, D.W. Doerfler, S.D. Hammond, K.S. Hemmert, M.A. Heroux, P.T. Lin, K.T. Pedretti, A.F. Rodrigues, T.G. Trucano, and J.P. Luitjens. Exascale design space exploration and co-design. *Future Generation Computer Systems*, 30(0):46–58, 2014. Special Issue on Extreme Scale Parallel Architectures and Systems, Cryptography in Cloud Computing and Recent Advances in Parallel and Distributed Systems, {ICPADS} 2012 Selected Papers .

[20] Christian Engelmann. Scaling to a million cores and beyond: Using light-weight simulation to understand the challenges ahead on the road to exascale. *Future Generation Computer Systems*, 30(0):59–65, 2014. Special Issue on Extreme Scale Parallel Architectures and Systems, Cryptography in Cloud Computing and Recent Advances in Parallel and Distributed Systems, {ICPADS} 2012 Selected Papers.

[21] Nikola Rajovic, Lluis Vilanova, Carlos Villavieja, Nikola Puzovic, and Alex Ramirez. The low power architecture approach towards exascale computing. *Journal of Computational Science*, 4(6):439–443, 2013. Scalable Algorithms for Large-Scale Systems Workshop (ScalA2011), Supercomputing 2011.

[22] N. Attig, P. Gibbon, and Th. Lippert. Trends in supercomputing: The European path to exascale. *Computer Physics Communications*, 182(9):2041–2046, 2011.